```
ISIS-II PL/M-80 V3.1 COMPILATION OF MODULE ERA
OBJECT MODULE PLACED IN ERASE.OBJ
COMPILER INVOKED BY:  PLM80 ERASE.PLM PAGEWIDTH(100) DEBUG OPTIMIZE
```

```
          $ TITLE('CP/M 3.0 --- ERA ')
          /* contains the confirm option */

  1       era:
          do;

          /*
            Copyright (C) 1982
            Digital Research
            P.O. Box 579
            Pacific Grove, CA 93950
          */

          /*
            Revised:
              19 Jan  80  by Thomas Rolander
              14 Sept 81  by Doug Huskey
              23 June 82  by John Knight
              03 Dec  82  by Bruce Skidmore
          */

  2  1    declare
             true    literally '1',
             false   literally '0',
             forever literally 'while true',
             lit     literally 'literally',
             proc    literally 'procedure',
             dcl     literally 'declare',
             addr    literally 'address',
             cr      literally '13',
             lf      literally '10',
             ctrlc   literally '3',
             ctrlx   literally '18h',
             tab     literally '9',
             bksp    literally '8',
             cpmversion     literally '30h',
             dcnt$offset    literally '45h',
             searcha$offset literally '47h',
             searchl$offset literally '49h',
             hashl$offset   literally '00h',
             hash2$offset   literally '02h',
             hash3$offset   literally '04h';

  3  1    declare plm label public;

          /*****************************************
          *                                        *
          *      B D O S   INTERFACE               *
          *                                        *
          *****************************************/
```

COPYRIGHT © 1982
DIGITAL RESEARCH
P. O. BOX 579
PACIFIC GROVE, CA 93950
SER. #

```
 4  1       mon1:
              procedure (func,info) external;
 5  2            declare func byte;
 6  2            declare info address;
 7  2           end mon1;

 8  1       mon2:
              procedure (func,info) byte external;
 9  2            declare func byte;
10  2            declare info address;
11  2           end mon2;

12  1       mon3:
              procedure (func,info) address external;
13  2            declare func byte;
14  2            declare info address;
15  2           end mon3;

16  1       parse:
              procedure (pfcb) address external;
17  2            declare pfcb address;
18  2           end parse;

19  1       declare cmdrv     byte    external;  /* command drive     */
20  1       declare fcb (1)   byte    external;  /* 1st default fcb   */
21  1       declare fcb16 (1) byte    external;  /* 2nd default fcb   */
22  1       declare pass0     address external;  /* 1st password ptr  */
23  1       declare len0      byte    external;  /* 1st passwd length */
24  1       declare pass1     address external;  /* 2nd password ptr  */
25  1       declare len1      byte    external;  /* 2nd passwd length */
26  1       declare tbuff (1) byte    external;  /* default dma buffer */


             /**************************************
              *                                    *
              *      B D O S   Externals           *
              *                                    *
              **************************************/

27  1       read$console:
              procedure byte;
28  2            return mon2 (1,0);
29  2           end read$console;


30  1       printchar:
              procedure(char);
31  2            declare char byte;
32  2            call mon1(2,char);
33  2           end printchar;

34  1       conin:
              procedure byte;
35  2            return mon2(6,0fdh);
36  2           end conin;
```

COPYRIGHT © 1982
DIGITAL RESEARCH
P. O. BOX 579
PACIFIC GROVE, CA 93950

SER. #_____

```
37  1       print$buf:
              procedure (buffer$address);
38  2           declare buffer$address address;
39  2           call mon1 (9,buffer$address);
40  2         end print$buf;


41  1       read$console$buf:
              procedure (buffer$address,max) byte;
42  2           declare buffer$address address;
43  2           declare new$max based buffer$address address;
44  2           declare max byte;
45  2           new$max = max;
46  2           call mon1(10,buffer$address);
47  2           buffer$address = buffer$address + 1;
48  2           return new$max;   /* actually number of chars input */
49  2       end read$console$buf;

50  1       check$con$stat:
              procedure byte;
51  2           return mon2 (11,0);
52  2         end check$con$stat;

53  1       version: procedure address;
              /* returns current cp/m version # */
54  2         return mon3(12,0);
55  2         end version;

56  1       setdma: procedure(dma);
57  2         declare dma address;
58  2         call mon1(26,dma);
59  2         end setdma;

60  1       search$first:
              procedure (fcb$address) byte;
61  2           declare fcb$address address;
62  2           return mon2 (17,fcb$address);
63  2         end search$first;

64  1       search$next:
              procedure byte;
65  2           return mon2 (18,0);
66  2         end search$next;

67  1       delete$file:
              procedure (fcb$address) address;
68  2           declare fcb$address address;
69  2           return mon3 (19,fcb$address);
70  2         end delete$file;

71  1       get$user$code:
              procedure byte;
72  2           return mon2 (32,0ffh);
73  2         end get$user$code;

            /* 0ff => return BDOS errors */
74  1       return$errors:
```

COPYRIGHT © 1982
DIGITAL RESEARCH
P. O. BOX 579
PACIFIC GROVE, CA 93950

SER. #_____

```
                  procedure;
 75   2             call mon1 (45,0ffh);
 76   2           end return$errors;

 77   1         declare scbpd structure
                  (offset byte,
                   set    byte,
                   value  address);

 78   1         getscbword:
                  procedure (offset) address;
 79   2           declare offset byte;
 80   2           scbpd.offset = offset;
 81   2           scbpd.set = 0;
 82   2           return mon3(49,.scbpd);
 83   2         end getscbword;

 84   1         setscbword:
                  procedure (offset,value);
 85   2           declare offset byte;
 86   2           declare value address;
 87   2           scbpd.offset = offset;
 88   2           scbpd.set = 0FEh;
 89   2           scbpd.value = value;
 90   2           call mon1(49,.scbpd);
 91   2         end setscbword;

 92   1         set$console$mode: procedure;
                /* set console mode to ctrl-c only */
 93   2           call mon1(109,1);
 94   2         end set$console$mode;

 95   1         declare
                  parse$fn structure (
                    buff$adr  address,
                    fcb$adr   address);


                /****************************************
                 *                                      *
                 *         GLOBAL VARIABLES             *
                 *                                      *
                 ****************************************/

 96   1         declare successful lit '0FFh';

 97   1         declare dir$entry$adr address;
 98   1         declare dir$entry based dir$entry$adr (1) byte;
 99   1         declare confirm$opt byte initial (false);
100   1         declare passwd$opt byte initial (false);
101   1         declare save$passwd (8) byte;
102   1         declare (savdcnt,savsearcha,savsearchl) address;
103   1         declare (hash1,hash2,hash3) address;

                /* options scanner variables and data */
104   1         declare
                  options(*) byte
```

COPYRIGHT © 1982
DIGITAL RESEARCH
P. O. BOX 579
PACIFIC GROVE, CA 93950
SER. #

```
            data('PASSWORDOCONFIRM',0ffh),

            off$opt(*) byte data(0,9,16),

            end$list    byte data (0ffh),

            delimiters(*) byte data (0,'[]=, ',0,0ffh),

            SPACE       byte data(5),

            j       byte initial(0),
            buf$ptr     address,
            index       byte,
            endbuf      byte,
            delimiter   byte;

105  1      declare end$of$string byte initial('0');


           /*****************************************
            *                                       *
            *      S U B R O U T I N E S            *
            *                                       *
            *****************************************/

          /* - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - */

          /*              * * *  Option scanner  * * *                    */


106  1     separator: procedure(character) byte;

                        /* determines if character is a
                           delimiter and which one */
107  2        declare k   byte,
                 character   byte;

108  2        k = 1;
109  2     loop:   if delimiters(k) = end$list then return(0);
111  2        if delimiters(k) = character then return(k);    /* null = 25 */
113  2           k = k + 1;
114  2           go to loop;

115  2     end separator;

116  1     opt$scanner:    procedure(list$ptr,off$ptr,idx$ptr);
                           /* scans the list pointed at by idxptr
                              for any strings that are in the
                              list pointed at by list$ptr.
                              Offptr points at an array that
                              contains the indices for the known
                              list. Idxptr points at the index
                              into the list. If the input string
                              is unrecognizable then the index is
                              0, otherwise > 0.

                           First, find the string in the known
```

COPYRIGHT © 1982
DIGITAL RESEARCH
P. O. BOX 579
PACIFIC GROVE, CA 93950
SER. #_____

```
                        list that starts with the same first
                        character.  Compare up until the next
                        delimiter on the input. if every input
                        character matches then check for
                        uniqueness.  Otherwise try to find
                        another known string that has its first
                        character match, and repeat.  If none
                        can be found then return invalid.

                        To test for uniqueness, start at the
                        next string in the knwon list and try
                        to get another match with the input.
                        If there is a match then return invalid.

                        else move pointer past delimiter and
                        return.

                    P.Balma     */

117  2      declare
               buff        based buf$ptr (1) byte,
               idx$ptr     address,
               off$ptr     address,
               list$ptr    address;

118  2      declare
               i       byte,
               j       byte,
               list        based list$ptr (1) byte,
               offsets     based off$ptr (1) byte,
               wrd$pos     byte,
               character   byte,
               letter$in$word  byte,
               found$first byte,
               start       byte,
               index       based idx$ptr byte,
               save$index  byte,
               (len$new,len$found) byte,
               valid       byte;

        /*****************************************************************************/
        /*          internal subroutines                                             */
        /*****************************************************************************/

119  2      check$in$list: procedure;
                        /* find known string that has a match with
                           input on the first character.  Set index
                           = invalid if none found.   */

120  3          declare i   byte;

121  3          i = start;
122  3          wrd$pos = offsets(i);
123  3          do while list(wrd$pos) <> end$list;
124  4              i = i + 1;
125  4              index = i;
126  4              if list(wrd$pos) = character then return;
```

COPYRIGHT © 1982
DIGITAL RESEARCH
P. O. BOX 579
PACIFIC GROVE, CA 93950
SER. #_____

```
128  4            wrd$pos = offsets(i);
129  4        end;
                      /* could not find character */
130  3        index = 0;
131  3        return;
132  3    end check$in$list;

133  2    setup:  procedure;
134  3        character = buff(0);
135  3        call check$in$list;
136  3        letter$in$word = wrd$pos;
                      /* even though no match may have occurred, position
                         to next input character.  */
137  3        i = 1;
138  3        character = buff(1);
139  3    end setup;

140  2    test$letter:    procedure;
                      /* test each letter in input and known string */

141  3        letter$in$word = letter$in$word + 1;

                            /* too many chars input? 0 means
                               past end of known string */
142  3        if list(letter$in$word) = end$of$string then valid = false;
             else
144  3        if list(letter$in$word) <> character then valid = false;

             i = i + 1;
147  3        character = buff(i);

148  3    end test$letter;

149  2    skip:   procedure;
                            /* scan past the offending string;
                               position buf$ptr to next string...
                               skip entire offending string;
                               ie., falseopt=mod, [note: comma or
                               space is considered to be group
                               delimiter] */
150  3        character = buff(i);
151  3        delimiter = separator(character);
             /* No skip for ERA */
152  3            do while ((delimiter < 1) or (delimiter > 6));
153  4            i = i + 1;
154  4            character = buff(i);
155  4            delimiter = separator(character);
156  4        end;
157  3        endbuf = i;
158  3        buf$ptr = buf$ptr + endbuf + 1;
159  3        return;
160  3    end skip;

161  2    eat$blanks: procedure;

162  3        declare charac  based buf$ptr byte;
```

CP/M plus Ver 3.0
COPYRIGHT © 1982
DIGITAL RESEARCH
P.O. BOX 579
PACIFIC GROVE, CA 93950
SER. # CP3-135

```
163  3          do while ((delimiter <> separator(charac)) = SPACE);
164  4              buf$ptr = buf$ptr + 1;
165  4          end;

166  3      end eat$blanks;

           /*****************************************************************************/
           /*          end of internals                                                 */
           /*****************************************************************************/


                              /* start of procedure */
167  2          call eat$blanks;
168  2          start = 0;
169  2          call setup;

                              /* match each character with the option
                                 for as many chars as input
                                 Please note that due to the array
                                 indices being relative to 0 and the
                                 use of index both as a validity flag
                                 and as a index into the option/mods
                                 list, index is forced to be +1 as an
                                 index into array and 0 as a flag*/

170  2          do while index <> 0;
171  3              start = index;
172  3              delimiter = separator(character);

                              /* check up to input delimiter */

173  3              valid = true;       /* test$letter resets this */
174  3              do while delimiter = 0;
175  4                  call test$letter;
176  4                  if not valid then go to exit1;
178  4                  delimiter = separator(character);
179  4              end;

180  3              go to good;

                              /* input <> this known string;
                                 get next known string that
                                 matches */
181  3      exit1:      call setup;
182  3          end;
                              /* fell through from above, did
                                 not find a good match*/
183  2          endbuf = i;          /* skip over string & return*/
184  2          call skip;
185  2          return;

                              /* is it a unique match in options
                                 list? */
186  2      good:   endbuf = i;
187  2          len$found = endbuf;
188  2          save$index = index;
```

COPYRIGHT © 1982
DIGITAL RESEARCH
P. O. BOX 579
PACIFIC GROVE, CA 93950
SER. #

```
189  2          valid = false;
190  2      next$opt:
                start = index;
191  2          call setup;
192  2          if index = 0 then go to finished;

                        /* look at other options and check
                           uniqueness */

194  2          len$new = offsets(index + 1) - offsets(index) - 1;
195  2          if len$new = len$found then do;
197  3              valid = true;
198  3              do j = 1 to len$found;
199  4                  call test$letter;
200  4                  if not valid then go to next$opt;
202  4              end;
203  3          end;
204  2          else go to nextopt;
                        /* fell through...found another valid
                           match --> ambiguous reference */
205  2        index = 0;
206  2        call skip;      /* skip input field to next delimiter*/
207  2        return;

208  2      finished:         /* unambiguous reference */
            index = save$index;
209  2      buf$ptr = buf$ptr + endbuf;
210  2      call eat$blanks;
211  2      if delimiter <> 0 then
212  2           buf$ptr = buf$ptr + 1;
               else
213  2           delimiter = 5;
214  2      return;

215  2      end opt$scanner;

        /* - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - */

216  1      break: procedure;
217  2        if check$con$stat then do;
219  3          call print$buf(.(cr,lf,'*** Aborted by ^C ***$'));
220  3          call mon1(0,0);
221  3        end;
222  2      end break;

        /* - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - */

                        /* upper case character from console */
223  1      crlf:   proc;
224  2          call printchar(cr);
225  2          call printchar(lf);
226  2          end crlf;

        /* - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - */

        /* fill string @ s for c bytes with f */
227  1      fill:   proc(s,f,c);
```

COPYRIGHT © 1982
DIGITAL RESEARCH
P. O. BOX 579
PACIFIC GROVE, CA 93950
SER. #

```
228  2        dcl s addr,
                  (f,c) byte,
                  a based s byte;

229  2           do while (c:=c-1)<>255;
230  3           a = f;
231  3           s = s+1;
232  3           end;
233  2        end fill;

         /* - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - */

                        /* error message routine */
234  1     error:  proc(code);
235  2       declare
             code byte;

236  2        call printchar(' ');
237  2        if code=1 then
238  2           call print$buf(.(cr,lf,'Disk I/O $'));
239  2        if code=2 then
240  2           call print$buf(.(cr,lf,'Drive $'));
241  2        if code = 3 or code = 2 then
242  2           call print$buf(.('Read Only$'));
243  2        if code = 5 then
244  2           call print$buf(.('Currently Opened$'));
245  2      . if code = 7 then
246  2           call print$buf(.('Password Error$'));
247  2        if code < 3 then
248  2           call mon1(0,0);
249  2        end error;

         /* - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - */

          /* try to delete fcb at fcb$address
                    return error code if unsuccessful */
250  1       delete:
            procedure(fcb$address) byte;
251  2        declare
              fcb$address address,
              fcbv based  fcb$address (32) byte,
              error$code  address,
              code        byte;

252  2        if passwd$opt then
253  2           fcbv(5) = fcbv(5) or 80h;
254  2        call setdma(.save$passwd(0));          /* password */
255  2        fcbv(0) = fcb(0);                      /* drive */
256  2        error$code = delete$file(fcb$address);
257  2        fcbv(5) = fcbv(5) and 7fh;             /* reset xfcb bit */
258  2        if low(error$code) = 0FFh then do;
260  3           code = high(error$code);
261  3           if (code=1) or (code=2) then
262  3              call error(code);
263  3           return code;
264  3           end;
265  2        return successful;
```

COPYRIGHT © 1982
DIGITAL RESEARCH
P. O. BOX 579
PACIFIC GROVE, CA 93950
SER. #

```
266  2        end delete;

         /* - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - */

                        /* upper case character from console */
267  1     ucase:  proc byte;
268  2        dcl c byte;

269  2        if (c:=conin) >= 'a' then
270  2           if c < '{' then
271  2              return(c-20h);
272  2        return c;
273  2        end ucase;

         /* - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - */

                        /* get password and place at fcb + 16 */
274  1     getpasswd:  proc;
275  2        dcl (i,c) byte;

276  2        call print$buf(.('Password: ','$'));
277  2     retry:
            call fill(.save$passwd(0),' ',8);
278  2           do i = 0 to 7;
279  3     nxtchr:
              if (c:=ucase) >= ' ' then
280  3                 save$passwd(i)=c;
281  3              if c = cr then
282  3                 go to exit;
283  3              if c = ctrlx then
284  3                 goto retry;
285  3              if c = bksp then do;
287  4                 if i<1 then
288  4                    goto retry;
289  4                 else do;
290  5                    save$passwd(i:=i-1)=' ';
291  5                    goto nxtchr;
292  5                    end;
293  4                 end;
294  3              if c = 3 then
295  3                 call mon1(0,0);
296  3              end;
297  2     exit:
            c = check$con$stat;           /* clear raw I/O mode */
298  2        end getpasswd;

         /* - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - */

                        /* error on deleting a file */
299  1        file$err: procedure(code);
300  2           declare code byte;

301  2           if not confirm$opt then do; /* print file */
303  3             call printchar('A'+fcb(0)-1);
304  3             call printchar(':');
305  3             call printchar(' ');
306  3             do k=1 to 11;
```

COPYRIGHT © 1982
DIGITAL RESEARCH
P. O. BOX 579
PACIFIC GROVE, CA 93950
SER. #

```
307  4              if k=9 then
308  4                call printchar(',');
309  4              call printchar(dir$entry(k));
310  4            end;
311  3            call print$buf(.(' $'));
312  3          end;
313  2          call print$buf(.('Not erased, $'));
314  2          call error(code);
315  2          call crlf;
316  2        end file$err;

         /* - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - */

317  1    erase: procedure;
318  2      if (code:=delete(.fcb)) <> successful then do;
320  3        if code < 3 then
321  3          call error(code);
322  3        else if code = 7 then do;
324  4          call file$err(code);
325  4          call getpasswd;
326  4          call crlf;
327  4          code = delete(.fcb);
328  4          end;
             if code <> successful then
330  3          call file$err(code);
331  3        end;
332  2    end erase;

         /* - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - */

333  1      parse$options: procedure;
334  2        declare
               t address,
               char based t byte,
               i byte;

335  2        delimiter = 1;
336  2        index = 0;
337  2        do while ((delimiter <> 0) and (delimiter <> 2) and (delimiter <> 6));
338  3          call opt$scanner(.options(0),.off$opt(0),.index);
339  3          if index = 0 then do;
                 /* unrecognized option */
341  4            call print$buf(.(cr,lf,'ERROR: Missing Delimiter or$'));
342  4            call print$buf(.(cr,lf,'       Unrecognized Option $'));
343  4            call print$buf(.('Near: $'));
344  4            t = buf$ptr - endbuf - 1;
345  4            do i = 1 to endbuf;
346  5              call printchar(char);
347  5              t = t + 1;
348  5            end;
349  4            call mon1(0,0);
350  4          end;
351  3          if index = 1 then
352  3            passwd$opt = true;
353  3          if index = 2 then
354  3            confirm$opt = true;
355  3        end;
```

COPYRIGHT © 1982
DIGITAL RESEARCH
P. O. BOX 579
PACIFIC GROVE, CA 93950
SER. #_____

```
356  2       end parse$options;

            /* - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - */

357  1      input$found: procedure (buffer$adr) byte;
358  2        declare buffer$adr address;
359  2        declare char based buffer$adr byte;
360  2        do while (char = ' ') or (char = tab);
361  3          buffer$adr = buffer$adr + 1;
362  3        end;
363  2        if char = 0 then  /* eoln */
364  2          return false;   /* input not found */
               else
365  2          return true;    /* input found */
366  2      end input$found;

            /****************************************
             *                                      *
             *      M A I N  P R O G R A M          *
             *                                      *
             ****************************************/

367  1      declare (i,k,code,response,user,dcnt) byte;
368  1      declare status address;
369  1      declare char$count byte;
370  1      declare last$dseg$byte byte
               initial (0);
371  1      declare no$chars byte;
372  1      declare n based status byte;

373  1      plm:
          do;
374  2          if (low(version) < cpmversion) or (high(version) = 1) then do;
376  3            call print$buf(.('Requires CP/M 3.0 $'));
377  3            call mon1(0,0);
378  3          end;
379  2          call set$console$mode;
380  2          if not input$found(.tbuff(1)) then do;
                 /* prompt for file */
382  3            confirm$opt = true; /* confirm, unless otherwise specified */
383  3            call print$buf(.('Enter filename: $'));
384  3            no$chars = read$console$buf(.tbuff(0),40);
385  3            char$count = no$chars + 2;
386  3            call print$buf(.(cr,lf,'$'));
387  3            tbuff(1) = ' '; /* blank out nc field */
388  3            tbuff(char$count) = 00h;    /* eoln marker set */
                 /* convert input string to upper case */
389  3            do i = 1 to char$count;
390  4              if tbuff(i+1) >= 'a' then
391  4                if tbuff(i+1) < '{' then
392  4                  tbuff(i+1) = tbuff(i+1) - 20h;
393  4            end;
394  3          end;
395  2          parse$fn.buff$adr = .tbuff(1);
396  2          parse$fn.fcb$adr = .fcb;
397  2          status = parse(.parse$fn);
398  2          if status = 0FFFFh then do;
```

COPYRIGHT © 1982
DIGITAL RESEARCH
P.O. BOX 579
PACIFIC GROVE, CA 93950
SER. #

```
400  3            call print$buf(.('ERROR: Invalid file name $'));
401  3            call mon1(0,0);
402  3          end;
403  2          if status <> 0 then do;   /* options must follow */
405  3            do while m = ' ';
406  4              status = status + 1;  /* skip over blank delimiters */
407  4            end;
408  3            buf$ptr = status + 1;   /* skip first delimiter */
409  3            call parse$options;
410  3          end;
411  2          if fcb(0) = 0 then
412  2            fcb(0) = low (mon2 (25,0)) + 1;
413  2          user = get$user$code;
414  2          call return$errors;
415  2          call move(8,.fcb16,.save$passwd(0));
416  2          if not confirm$opt then do;
418  3            i = 0;
419  3            do while fcb(i:=i+1) = '?';
420  4            end;
421  3            if i > 11 then
422  3              if not passwd$opt then do;
424  4                call print$buf(.('Confirm delete all user files (Y/N)?$'));
425  4                response = read$console;
426  4                if not ((response = 'y') or (response = 'Y')) then
427  4                  call mon1(0,0);
428  4                call crlf;
429  4            end;
430  3          end;
431  2          call move(16,.fcb,.fcb16);
432  2          call setdma(.tbuff);
433  2          dcnt = search$first (.fcb16);
434  2          if dcnt = 0FFh then do;
436  3            call print$buf(.('No File $'));
437  3            call mon1(0,0);
438  3          end;
439  2          do while dcnt <> 0ffh;
440  3            dir$entry$adr = .tbuff(ror(dcnt,3) and 110$0000b);
441  3            savdcnt = getscbword(dcnt$offset);
442  3            savsearcha = getscbword(searcha$offset);
443  3            savsearchl = getscbword(searchl$offset);
                  /* save searched fcb's hash code (5 bytes) */
444  3            hash1 = getscbword(hash1$offset);
445  3            hash2 = getscbword(hash2$offset);
446  3            hash3 = getscbword(hash3$offset);
447  3            if confirm$opt then do;
449  4              if dir$entry(0) = user then do;
451  5                call printchar ('A'+fcb(0)-1);
452  5                call printchar (':');
453  5                call printchar (' ');
454  5                do k = 1 to 11;
455  6                  if k = 9
                         then call printchar (',');
457  6                  call printchar (dir$entry(k));
458  6                end;
459  5                call print$buf(.(' (Y/N)? $'));
460  5                response = read$console;
461  5                call printchar (cr);
```

COPYRIGHT © 1982
DIGITAL RESEARCH
P.O. BOX 579
PACIFIC GROVE, CA 93950
SER. #_____

```
 462  5                call printchar (lf);
 463  5                if response = ctrlc then do;
 465  6                  call print$buf(.(cr,lf,'*** Aborted by ^C ***$'));
 466  6                  call mon1(0,0);
 467  6                end;
 468  5                if (response = 'y') or
                          (response = 'Y') then do;
 470  6                    call move (12,.dir$entry(1),.fcb(1));
 471  6                    call erase;
 472  6                end;
 473  5              end;
 474  4            end;
 475  3            else do;    /* not confirm option */
 476  4              call move(12,.dir$entry(1),.fcb(1));
 477  4              call break;
 478  4              call erase;
 479  4            end;
 480  3            call setdma(.tbuff);
 481  3            call setscbword(dcnt$offset,savdcnt);
 482  3            call setscbword(searcha$offset,savsearcha);
 483  3            call setscbword(search1$offset,savsearch1);
                    /* restore hash code */
 484  3            call setscbword(hash1$offset,hash1);
 485  3            call setscbword(hash2$offset,hash2);
 486  3            call setscbword(hash3$offset,hash3);
 487  3            if .fcb16 <> savsearcha then    /* restore search fcb if destroyed */
 488  3              call move(16,.fcb16,savsearcha);
 489  3            dcnt = search$next;
 490  3          end;
 491  2          call mon1(0,0);
 492  2        end;
 493  1      end era;
```

COPYRIGHT © 1982
DIGITAL RESEARCH
P.O. BOX 579
PACIFIC GROVE, CA 93950
SER. #

MODULE INFORMATION:

```
    CODE AREA SIZE     = 0BB5H   2997D
    VARIABLE AREA SIZE = 0069H    105D
    MAXIMUM STACK SIZE = 000AH     10D
    824 LINES READ
    0 PROGRAM ERROR(S)
```

END OF PL/M-80 COMPILATION

```
0000 ERASE
0000 MCD80A
0004 BDISK     0080 BUFF      0080 BUFFA     0050 CMDRV     0000 CPU
007C CR              006D DOLLA     005C FCB       006C FCB16     005C FCBA
005C IFCB      005C IFCBA     0003 IOBYTE    0053 LEN0      0056 LEN1
0006 MAXB      0006 MEMSIZ    0005 MON1      0005 MON2      0005 MON2A
0005 MON3      0000 OFFSET    006E PARMA     0051 PASS0     0054 PASS1
007F RO              007D RR        007D RRECA     007F RRECO     005C SFCB
0080 TBUFF
0000 ERA
0F80 MEMORY    02E6 PLM       0640 READCONSOLE              0649 PRINTCHAR
0F17 CHAR      0659 CONIN     0662 PRINTBUF  0F18 BUFFERADDRESS
0672 READCONSOLEBUF           0F1A BUFFERADDRESS            0F1C MAX
0699 CHECKCONSTAT             06A2 VERSION   06AB SETDMA     0F1D DMA
06BB SEARCHFIRST              0F1F FCBADDRESS 06CB SEARCHNEXT 06D4 DELETEFILE
0F21 FCBADDRESS       06E4 GETUSERCODE             06ED RETURNERRORS
0F23 SCBPD     06F6 GETSCBWORD 0F27 OFFSET    070E SETSCBWORD 0F28 OFFSET
0F29 VALUE     0734 SETCONSOLEMODE           0F2B PARSEFN
0F2F DIRENTRYADR              0F31 CONFIRMOPT 0F32 PASSWDOPT  0F33 SAVEPASSWD
0F3B SAVDCNT   0F3D SAVSEARCHA 0F3F SAVSEARCHL 0F41 HASH1     0F43 HASH2
0F45 HASH3     0180 OPTIONS   0191 OFFOPT    0194 ENDLIST   0195 DELIMITERS
019D SPACE     0F47 J         0F48 BUFPTR    0F4A INDEX     0F4B ENDBUF
0F4C DELIMITER 0F4D ENDOFSTRING              073D SEPARATOR  0F4E CHARACTER
0F4F K         0746 LOOP      0774 OPTSCANNER 0F50 LISTPTR   0F52 OFFPTR
0F54 IDXPTR    0F56 I         0F57 J         0F58 WRDPOS    0F59 CHARACTER
0F5A LETTERINWORD             0F5B FOUNDFIRST 0F5C START     0F5D SAVEINDEX
0F5E LENNEW    0F5F LENFOUND  0F60 VALID     089A CHECKINLIST
0F61 I         08F0 SETUP     090E TESTLETTER 094E SKIP      09AA EATBLANKS
07D4 EXIT1     07E4 GOOD      07F9 NEXTOPT   086A FINISHED  09C6 BREAK
09DC CRLF      09E7 FILL      0F62 S         0F64 F         0F65 C
0A12 ERROR     0F66 CODE      0A82 DELETE    0F67 FCBADDRESS 0F69 ERRORCODE
0F6B CODE      0AF0 UCASE     0F6C C         0B0D GETPASSWD  0F6D I
0F6E C         0B13 RETRY     0B2C NXTCHR    0B97 EXIT      0B9E FILEERR
0F6F CODE      0C04 ERASE     0C52 PARSEOPTIONS             0F70 T
0F72 I         0CFE INPUTFOUND 0F73 BUFFERADR 0F75 I         0F76 K
0F77 CODE      0F78 RESPONSE  0F79 USER      0F7A DCNT      0F7B STATUS
0F7D CHARCOUNT 0F7E LASTDSEGBYTE             0F7F NOCHARS
0000 PARSE
0E7E DBLNK1    0E72 DEBLNK    0E30 DELIM     0E6B DELIM1    0E6E DELIM2
0E28 PADFLD    0DC6 PARS81    0DC7 PARS82    0D35 PARSE     0D56 PARSE1
0D84 PARSE2    0D87 PARSE3    0D99 PARSE4    0D9D PARSE5    0D9F PARSE6
0DB3 PARSE7    0DB7 PARSE8    0DD8 PARSE9    0E01 PWFLD     0E14 PWFLD1
0DF3 SETFD1    0DF5 SETFD2    0DF9 SETFD3    0DDE SETFLD
```

COPYRIGHT © 1982
DIGITAL RESEARCH
P.O. BOX 579
PACIFIC GROVE, CA 93950

SER. #_____

```
0000 ERASE$
0000 MCD80A$
0000 ERA$
0640 27         0640 28         0649 29         0649 30         064D 32
0658 33         0659 34         0659 35         0662 36         0662 37
0668 39         0671 40         0672 41         067A 45         0684 46
068D 47         0694 48         0699 49         0699 50         0699 51
06A2 52         06A2 53         06A2 54         06AB 55         06AB 56
06B1 58         06BA 59         06BB 60         06C1 62         06CB 63
06CB 64         06CB 65         06D4 66         06D4 67         06DA 69
06E4 70         06E4 71         06E4 72         06ED 73         06ED 74
06ED 75         06F5 76         06F6 78         06FA 80         0700 81
0705 82         070E 83         070E 84         0716 87         071C 88
0721 89         072B 90         0733 91         0734 92         0734 93
073C 94         073D 106        0741 108        0746 109        0755 110
0758 111        0768 112        076C 113        0770 114        0773 115
0774 116        089A 119        089A 121        08A0 122        08AB 123
08BC 124        08C0 125        08C7 126        08D8 127        08D9 128
08E7 129        08EA 130        08EF 131        08F0 132        08F0 133
08F0 134        08F7 135        08FA 136        0900 137        0905 138
090D 139        090E 140        090E 141        0912 142        0920 143
0928 144        0939 145        093E 146        0942 147        094D 148
094E 149        094E 150        095C 151        0966 152        097B 153
097F 154        098A 155        0994 156        0997 157        099D 158
09A9 159        09AA 160        09AA 161        09AA 163        09BB 164
09C2 165        09C5 166        0785 167        0788 168        078D 169
0790 170        0799 171        07A0 172        07AA 173        07AF 174
07B7 175        07BA 176        07C1 177        07C4 178        07CE 179
07D1 180        07D4 181        07D7 182        07DA 183        07E0 184
07E3 185        07E4 186        07EA 187        07ED 188        07F4 189
07F9 190        0800 191        0803 192        080C 193        080F 194
082B 195        0833 197        0838 198        0847 199        084A 200
0851 201        0854 202        085B 203        085E 204        0861 205
0866 206        0869 207        086A 208        0871 209        087F 210
0882 211        088A 212        0894 213        0899 214        089A 215
09C6 216        09C6 217        09CD 219        09D3 220        09DB 221
09DB 222        09DC 223        09DC 224        09E1 225        09E6 226
09E7 227        09F4 229        0A00 230        0A07 231        0A0E 232
0A11 233        0A12 234        0A16 236        0A1B 237        0A23 238
0A29 239        0A31 240        0A37 241        0A4F 242        0A55 243
0A5D 244        0A63 245        0A6B 246        0A71 247        0A79 248
0A81 249        0A82 250        0A88 252        0A8F 253        0A9A 254
0AA0 255        0AA7 256        0AB2 257        0ABD 258        0AC6 260
0ACD 261        0AE2 262        0AE9 263        0AED 264        0AED 265
0AF0 266        0AF0 267        0AF0 269        0AFB 270        0B03 271
0B09 272        0B0D 273        0B0D 274        0B0D 276        0B13 277
0B1E 278        0B2C 279        0B37 280        0B44 281        0B4C 282
0B4F 283        0B57 284        0B5A 285        0B62 287        0B6A 288
0B6D 290        0B7D 291        0B80 292        0B80 293        0B80 294
0B88 295        0B90 296        0B97 297        0B9D 298        0B9E 299
0BA2 301        0BA9 303        0BB3 304        0BB8 305        0BBD 306
0BCB 307        0BD3 308        0BD8 309        0BE6 310        0BED 311
0BF3 312        0BF3 313        0BF9 314        0C00 315        0C03 316
0C04 317        0C04 318        0C12 320        0C1A 321        0C24 322
0C2C 324        0C33 325        0C36 326        0C39 327        0C42 328
0C42 329        0C4A 330        0C51 331        0C51 332        0C52 333
0C52 335        0C57 336        0C5C 337        0C80 338        0C8D 339
0C95 341        0C9B 342        0CA1 343        0CA7 344        0CB4 345
0CC3 346        0CCA 347        0CD1 348        0CD8 349        0CE0 350
0CE0 351        0CE8 352        0CED 353        0CF5 354        0CFA 355
0CFD 356        0CFE 357        0D04 360        0D1B 361        0D22 362
0D25 363        0D2E 364        0D31 365        0D34 366        02E3 373
02E9 374        0301 376        0307 377        030F 378        030F 379
```

COPYRIGHT © 1982
DIGITAL RESEARCH
P. O. BOX 579
PACIFIC GROVE, CA 93950

SER. #_____

```
0312 380    031C 382    0321 383    0327 384    0332 385
033A 386    0340 387    0345 388    0350 389    035F 390
036E 391    037D 392    038A 393    0391 394    0391 395
0397 396    039D 397    03A6 398    03B3 400    03B9 401
03C1 402    03C1 403    03CD 405    03D6 406    03DD 407
03E0 408    03E7 409    03EA 410    03EA 411    03F2 412
0402 413    0408 414    040B 415    041B 416    0423 418
0428 419    043C 420    043F 421    0448 422    0450 424
0456 425    045C 426    0475 427    047D 428    0480 429
0480 430    0480 431    0490 432    0496 433    049F 434
04A7 436    04AD 437    04B5 438    04B5 439    04BD 440
04CF 441    04D7 442    04DF 443    04E7 444    04EF 445
04F7 446    04FF 447    0506 449    0510 451    051A 452
051F 453    0524 454    0532 455    053A 456    053F 457
054D 458    0557 459    055D 460    0563 461    0568 462
056D 463    0575 465    057B 466    0583 467    0583 468
059B 470    05B0 471    05B3 472    05B3 473    05B3 474
05B6 476    05CB 477    05CE 478    05D1 479    05D1 480
05D7 481    05E0 482    05E9 483    05F2 484    05FB 485
0604 486    060D 487    061A 488    062D 489    0633 490
0636 491    063E 493
0000 PARSE#
```

COPYRIGHT © 1982
DIGITAL RESEARCH
P. O. BOX 579
PACIFIC GROVE, CA 93950

SER. # _____